College of Information Technology, Western Governors University

Random Forest Regression on Flight Ticket Dataset

Report and Executive Summary

Luke Dorsett

D214 Data Ananytics Capstone

3/31/2024

Random Forest Regression on Flight Ticket Dataset

Report and Executive Summary

Research Question and Hypothesis

The airline industry is used by millions each year, yet it is often difficult to evaluate how expensive a ticket will be when preparing to purchase one online. In addition, the Indian aviation industry is projected to be the largest in the world by 2030 (Joshi, n.d.). Understanding the factors contributing to the price changes can help consumers purchase tickets at the best time, or at least better understand how expensive tickets will be. In the past, research has identified that the optimal model for this would be a Random Forest Regressor model, and that the number of days prior to the flight is a key factor in predicting the ticket price (Abdella et. al., 2019). With this knowledge, this study will answer the question if a Random Forest Regressor model can be created on the research dataset of Indian airline ticket prices. The null hypothesis for this study is that a Random Forest Regressor model cannot be made on the research dataset. The alternative hypothesis is that a Random Forest Regressor model can be made at <15% MAPE. If the analysis supports the alternative hypothesis, then consumers and businesses will more easily be able to understand the pricing tactics and systems implemented by airlines.

Data Collection

The dataset used is from Kaggle.com, by the user Shubham Bathwal, and contains 300,152 observations of Indian flights with their corresponding ticket price. The dataset is a combination of business and economy class tickets from the site EaseMyTrip.com between the dates February 11, 2022, and March 31, 2022. The overall sparsity of this dataset prior to cleaning and transformations is 0%. The dataset was already cleaned to remove null, missing, or

incorrect values. One major advantage of this data collection process is the ease of use of Kaggle.com, as the distribution, data types, and other attributes of the dataset were visible before downloading. A disadvantage of this was the size of the dataset chosen, which meant that many operations took hours or days to run given the hardware limitations present. This was partially circumvented by using Google Colab, as multiple operations could be run simultaneously without reducing the speed of either training.

Data extraction

Using Python and Pandas, the data was loaded in, and an extra index column was removed as it was not needed for the analysis. Pandas is the best Python library for dataset processing but is slightly slower by comparison to R and SAS (Brittain et. al. 2018).

```
data = pd.read_csv('Clean_Dataset.csv', sep=',')

data = data.iloc[:, 1:]

data.head()
```

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

The only additional cleaning was encoding the following categorical variables as multiple binary features using Pandas:

- Airline
- Flight
- Source_city
- Departure_time
- Arrival_time
- Destination_city
- Class
- Stops

Python is used as it is the optimal choice out of the data processing languages for dataset analysis and is the most widely used by data analytics professionals (Colliau et. al. 2017). Python is extremely intuitive and easy to use, but the downside is that the processing speeds are slower than other languages.

Analysis

The first step of analysis is to evaluate the distribution of the data using a Shapiro-Wilk test. This was done using the shapiro function from the library Scipy as seen below:

```
shapiro(data['price'])
✓ 0.0s
ShapiroResult(statistic=0.7521858811378479, pvalue=0.0)
```

The shapiro function is immensely useful, as the p value output reflects the normality of the data distribution. In this case, since the p value is below 0.05, the distribution is considered normal, meaning that there are no worries such as extreme outliers and uneven distributions which would cause problems when creating a model. A disadvantage to the shapiro function seen here is that the function does not always reflect accurate results with extremely large datasets. The data was then split randomly into 80% train, 20% test using the train_test_split function from the library sklearn as seen here:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
✓ 0.6s
```

Now that the data was split, the next step was to perform linear regression, and evaluate the impact of each feature on the target variable. This was conducted using the OLS function from statsmodels.api library. The OLS function is helpful due to the detailed summary of the regression it provides as seen below:

```
lr_X_train = add_constant(X_train)
lr = OLS(y_train, lr_X_train)
result = lr.fit()
result.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.925
Model:	OLS	**Adj. R-squared:**	0.925
Method:	Least Squares	**F-statistic:**	3557.
Date:	Fri, 29 Mar 2024	**Prob (F-statistic):**	0.00
Time:	07:08:30	**Log-Likelihood:**	-2.4379e+06
No. Observations:	240122	**AIC:**	4.877e+06
Df Residuals:	239288	**BIC:**	4.886e+06
Df Model:	833		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.12e+04	128.786	397.587	0.000	5.1e+04	5.15e+04
duration	58.0480	2.587	22.438	0.000	52.978	63.118
days_left	-128.3801	0.948	-135.400	0.000	-130.238	-126.522
airline_Air_India	-1365.0340	113.144	-12.065	0.000	-1586.794	-1143.274
airline_GO_FIRST	1153.3701	120.779	9.549	0.000	916.647	1390.093
airline_Indigo	1869.7451	105.864	17.662	0.000	1662.254	2077.237
airline_SpiceJet	3696.8292	177.649	20.810	0.000	3348.641	4045.017
airline_Vistara	2911.1010	126.589	22.997	0.000	2662.991	3159.212
flight_6E-113	-3082.1509	1056.161	-2.918	0.004	-5152.198	-1012.104
flight_6E-126	3884.6196	715.413	5.430	0.000	2482.429	5286.810
flight_6E-131	-2659.0511	755.380	-3.520	0.000	-4139.576	-1178.526
flight_6E-132	3979.8383	1056.957	3.765	0.000	1908.230	6051.447
flight_6E-135	-4970.2268	518.495	-9.586	0.000	-5986.463	-3953.991
flight_6E-136	-3248.9920	578.797	-5.613	0.000	-4383.419	-2114.565
flight_6E-146	6819.3410	1666.098	4.093	0.000	3553.832	1.01e+04

The The flight feature had many more unique values than expected, and the dataset feature size had expanded to over 1,000 features after encoding. Many of these columns were not statistically significant, as their p values were greater than 0.05, so they were removed from the dataset to prevent confusion within the dataset. The following code shows the features which were removed (over 700 in total removed):

```
[14] p_values = result.pvalues

     high_p_values = p_values[p_values > 0.05].index.tolist()

     print("Features with p values greater than 0.05:", high_p_values)

     Features with p values greater than 0.05: ['flight_6E-164', 'flight_6E-266', 'flight_6E-533', 'flight_6E-6515', 'flight_AI-424', 'flight_AI-483', 'fligh
```

The rest of the variables with p values below 0.05 were kept, as they were considered significant and that they all contribute meaningfully to predicting the target variable. Next, the model was created, and several hyperparameters were tested using sklearn and Bayesian Optimization for hyperparameter tuning. Several test models were trained to evaluate training time estimates, and then 4 models were created using recommended hyperparameters from Bayesian Optimization. The best model was chosen due to it having the lowest MAPE score. Random Forest Regression using Sklearn is useful as it is easy to implement and test different metrics with MAPE. A disadvantage of using Random Forest Regression is that there is great potential for overfitting. This is a problem Random Forest models encounter, especially on datasets this large (Xu et. al. 2007), but careful hyperparameter tuning and training practices were used to ensure this problem was mitigated as best as possible. The code for training and evaluating the models is seen below (final model results shown):

```
[37] rf_model.fit(X_train, y_train)
     ✓ 1652m 15.8s

...  RandomForestRegressor(max_depth=8, max_leaf_nodes=80, min_samples_split=20,
                           n_estimators=40, n_jobs=-1)

[38] y_pred = rf_model.predict(X_test)
     ✓ 2.7s

[39] mean_absolute_percentage_error(y_test, y_pred)
     ✓ 0.0s

...  0.1272164639390422
```

Bayesian Hyperparameter tuning is an effective method of finding optimal hyperparameters without having to try every possible combination or trying random combinations. The code and output of one of the training sessions using the Bayesian_Optimization library is below:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from bayes_opt import BayesianOptimization
import numpy as np

def mape_scorer(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape_score = make_scorer(mape_scorer, greater_is_better=False)

param_space = {
    'n_estimators': (20, 40),
    'max_depth': (8, 60),
    'min_samples_split': (2, 20),
    'min_samples_leaf': (1, 5),
    'max_leaf_nodes': (8, 80)
}
```

```
def objective_function(n_estimators, max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes):
    model = RandomForestRegressor(
        n_estimators=int(n_estimators),
        max_depth=int(max_depth),
        min_samples_split=int(min_samples_split),
        min_samples_leaf=int(min_samples_leaf),
        max_leaf_nodes=int(max_leaf_nodes),
        n_jobs=-1
    )
    score = cross_val_score(model, X_train, y_train, cv=5, scoring=mape_score).mean()
    return score

optimizer = BayesianOptimization(f=objective_function, pbounds=param_space, random_state=1)

optimizer.maximize(init_points=2, n_iter=10)

best_params = optimizer.max['params']
print(best_params)
```

Output:

iter	target	max_depth	max_le...	min_sa...	min_sa...	n_esti...
1	-23.8	29.69	59.86	1.0	7.442	22.94
2	-26.59	12.8	21.41	2.382	9.142	30.78
3	-23.79	29.76	59.29	1.773	8.314	22.77
4	-22.5	55.32	80.0	5.0	20.0	20.0
5	-23.75	60.0	60.68	5.0	20.0	40.0
6	-20.28	8.0	80.0	5.0	20.0	20.0
7	-20.28	8.0	80.0	5.0	20.0	40.0
8	-20.29	8.0	80.0	5.0	2.0	40.0
9	-20.3	8.0	80.0	5.0	2.0	20.0
10	-20.28	8.0	80.0	5.0	10.99	30.05

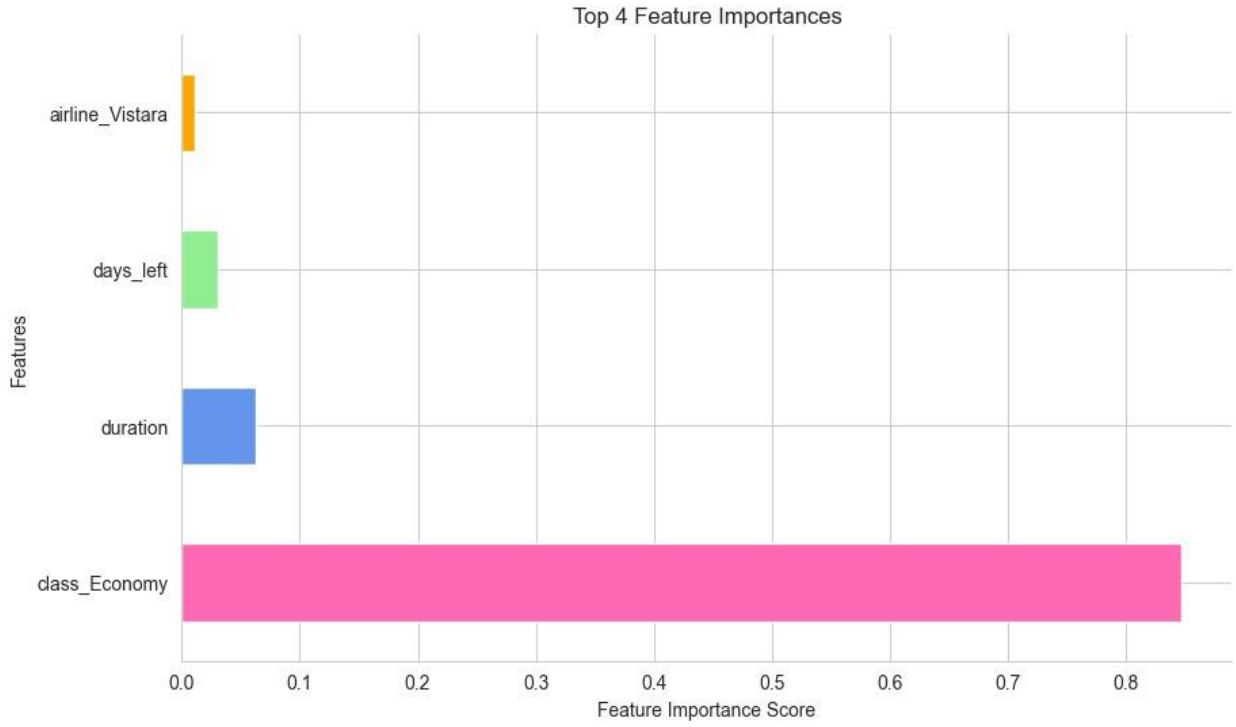
This process had advantages and disadvantages. Using these did save time and prevent numerous models trained with disappointing predictive capabilities, but two additional models were trained which had extremely disappointing test scores, despite the algorithm suggesting otherwise. However, it is known that not all models work well with Bayesian Optimization (Dewancker et. al., 2016), so that is likely the cause of the problem.

Implications

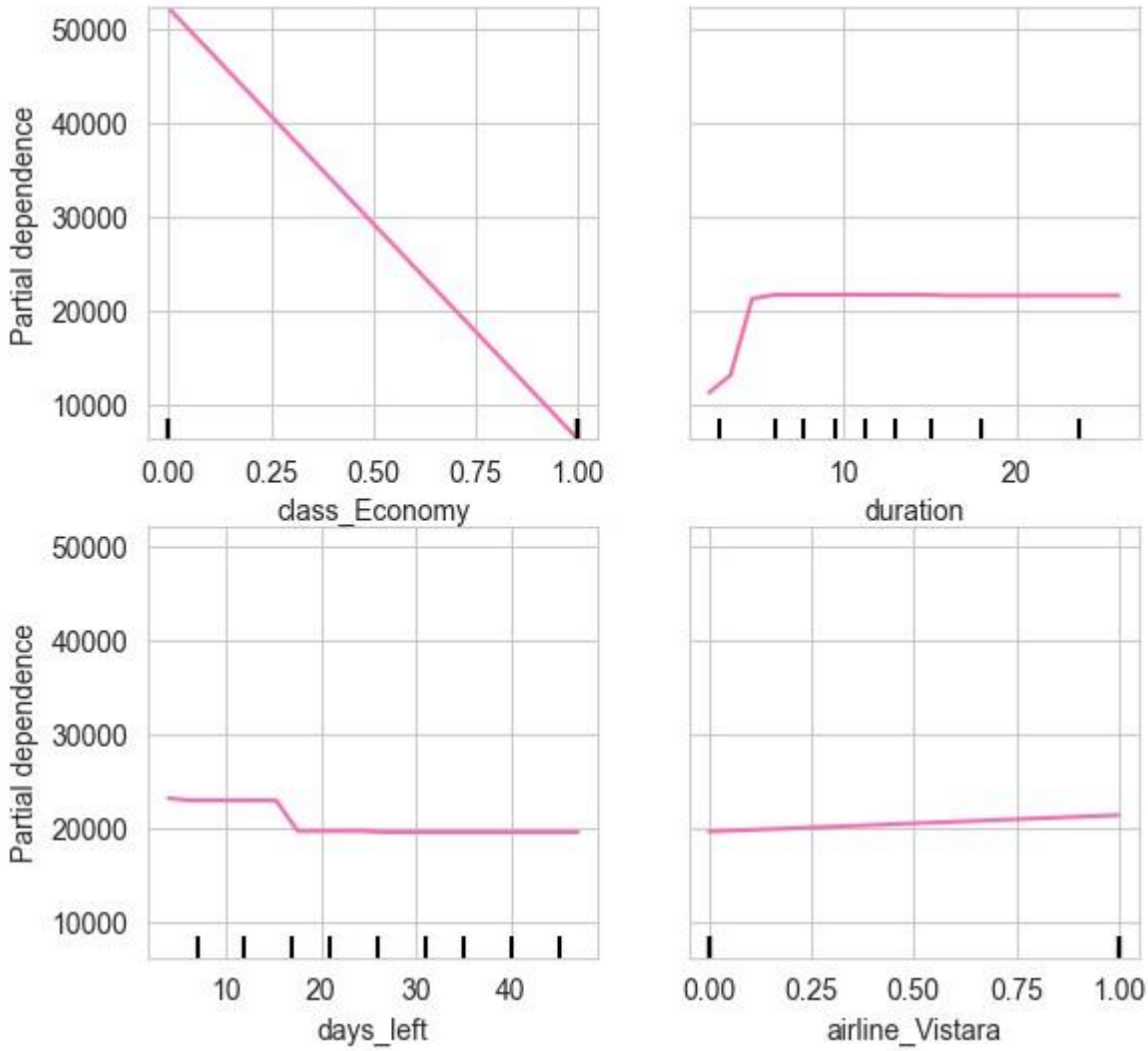
The best model, having MAPE of 12.72, is below 15%, so the null hypothesis is rejected since a model was successfully created of the research dataset. The following table shows the models trained with the respective MAPE scores:

Model No.	N_estimators	Max_depth	Max_leaf_nodes	MAPE
1	10	4	4	30.9%
2	20	8	8	24.56%
3	34	8	8	18.24%
4	40	8	80	12.72%

Additionally, each model also used hyperparameters of min_samples_leaf = 1, and min_samples_split = 20. The feature importance of the final model was observed, shown through the graph below:



The binary feature which identifies the ticket as economy or business class was by far the most important variable, which is understandable since business class tickets would be much more expensive on average. Flight duration, days before ticket purchasing, and the airline Vistara were the next 3 important features, but nowhere near the importance of the economy class identifier feature. Finally, these 4 features were examined using partial dependency plots. These show how the variable changes alongside the predicted Y value.



The most complex relationship out of the 4 is duration. As flights become longer, price sharply increases until about 5 hours is reached. Then, the target variable plateaus, indicating flights over 5 hours are comparable prices to that of 10- or 20-hour flights.

Limitations and Course of Action

A major limitation of this study was the hardware used. More complex models could not be created, as it would have taken weeks to complete, even using all CPU cores for training. Despite this, it is shown that a Random Forest Regressor model can accurately predict ticket

prices for Indian flights and provide valuable insights for customers, airlines, and other businesses. Given this null hypothesis was rejected, and a Random Forest Regressor can predict ticket prices, a course of action is for businesses and consumers to refine this model and apply it to better understand the pricing systems airlines use. A possible direction for future studies would be to utilize more advanced machine learning algorithms with the help of hardware which could parallelize the training of the models to create them in a reasonable time frame. These models would likely predict the data more accurately, as the number of predictors in each model would be much higher. Another direction for future studies would be to collect data from around the whole year in India, so that the dataset would be better generalized. Around a major holiday or travel season, this model would likely have much greater error due to the increased ticket prices. Overall, the results from this analysis demonstrate effective use of machine learning techniques for this subject and provide a foundation for further research into this topic.

Sources

Abdella, J. A., Zaki, N., Shuaib, K., & Khan, F. (2019). Airline ticket price and demand prediction: A survey. *Journal of King Saud University - Computer and Information Sciences*, *33*(4). https://doi.org/10.1016/j.jksuci.2019.02.001

Achyut Joshi. (n.d.). Achyut Joshi. Retrieved March 12, 2023, from https://achyutjoshi.github.io/btp/flightprices

Brittain, J., Cendon, M., Nizzi, J., & Pleis, J. (2018). Data Scientist's Analysis Toolbox: Comparison of Python, R, and SAS Performance. *SMU Data Science Review*, *1*(2). Retrieved March 7, 2023, from https://scholar.smu.edu/cgi/viewcontent.cgi?article=1021&context=datasciencereview

Colliau, T., Rogers, G., Hughes, Z., Ozgur, C., Hughes, Z., Bennie, E., & Myer-Tyson, quot; (2017). MatLab vs. Python vs. R MatLab vs. Python vs. R. *Journal of Data Science*, *15*, 355–372. Retrieved March 7, 2023, from https://scholar.valpo.edu/cgi/viewcontent.cgi?article=1049&context=cba_fac_pub

Dewancker, I., McCourt, M., & Clark, S. (2016, December 14). *Bayesian Optimization for Machine Learning : A Practical Guidebook*. ArXiv.org. Retrieved March 20, 2023, https://doi.org/10.48550/arXiv.1612.04858

Xu, P., & Jelinek, F. (2007). Random forests and the data sparseness problem in language modeling. *Computer Speech & Language*, *21*(1), 105–152. Retrieved March 7, 2023, from https://doi.org/10.1016/j.csl.2006.01.003